Ruby as a Node

Artur Sulej

Artur Sulej @ Fresha.com

/artur-sulej

/artursulej

Fresha.com

1. Big scale
2. Great DevEx
3. Solid Architecture

Agenda

Elixir  Ruby

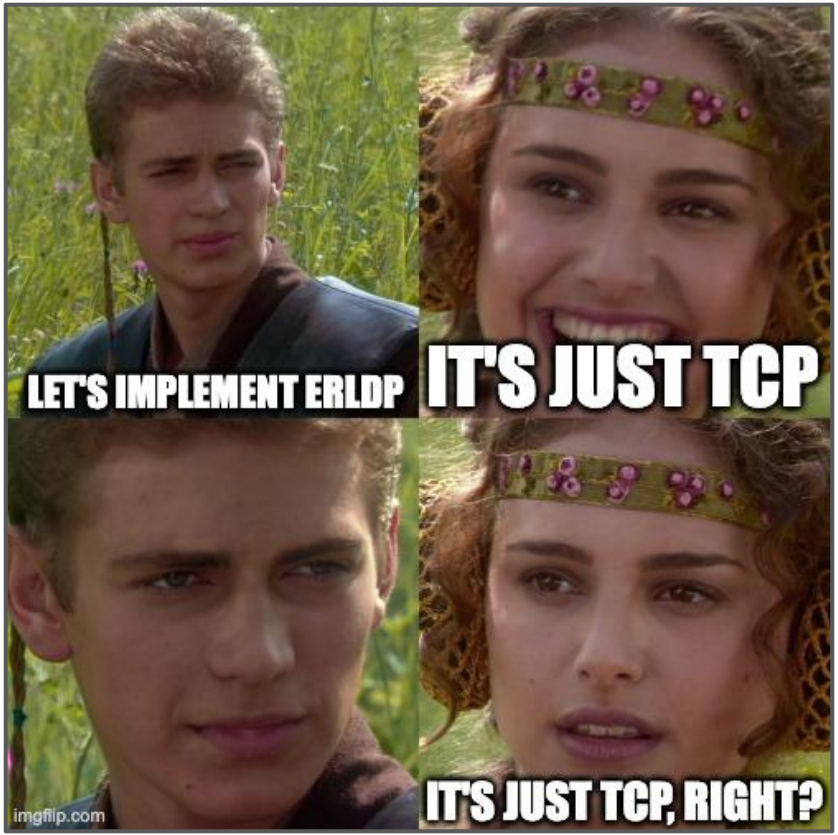
- Erlang Distribution Protocol in Ruby
- Elixir proxy with Port

ErlDP – recap

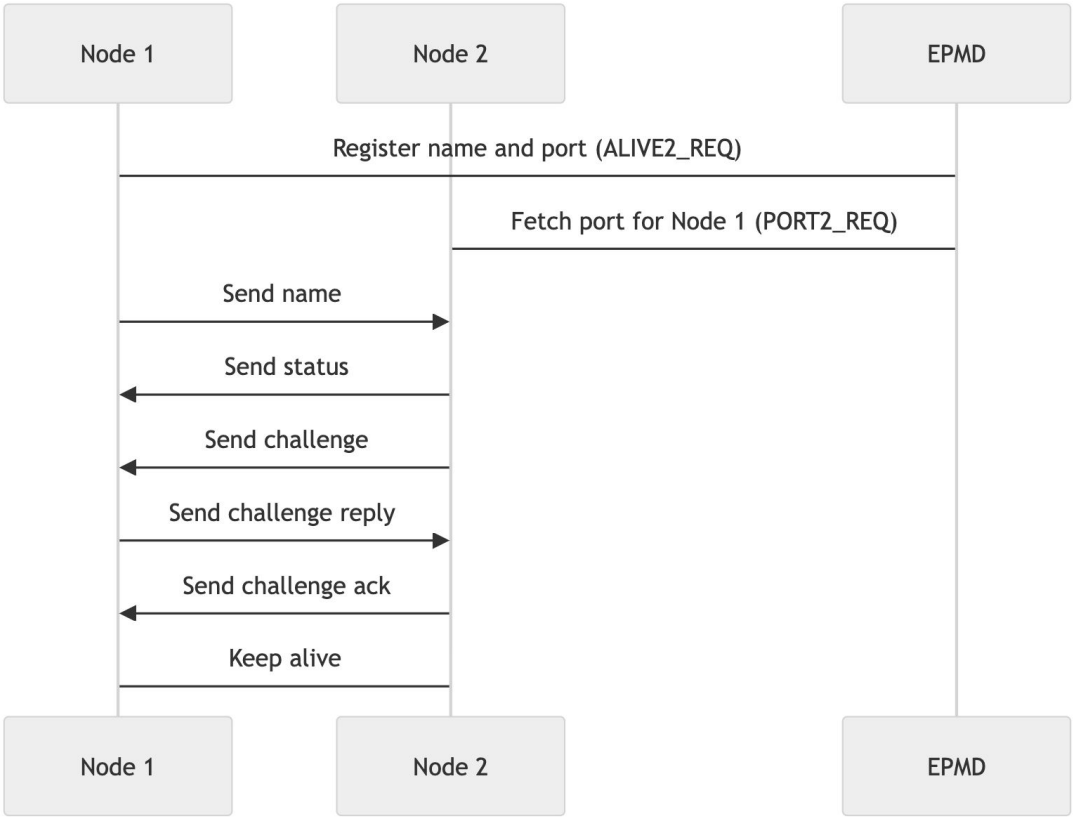
Erlang Distribution Protocol (ErlDP)

- Based on TCP
- Communication between distributed nodes mimics local communication within a single node
- Connecting nodes – handshake
- Erlang Port Mapper Daemon (EPMD) – keeps names and ports of nodes on a given host

ErlDP


LET'S IMPLEMENT ERLDP
IT'S JUST TCP
IT'S JUST TCP, RIGHT?
imgflip.com

ErlDP – handshake

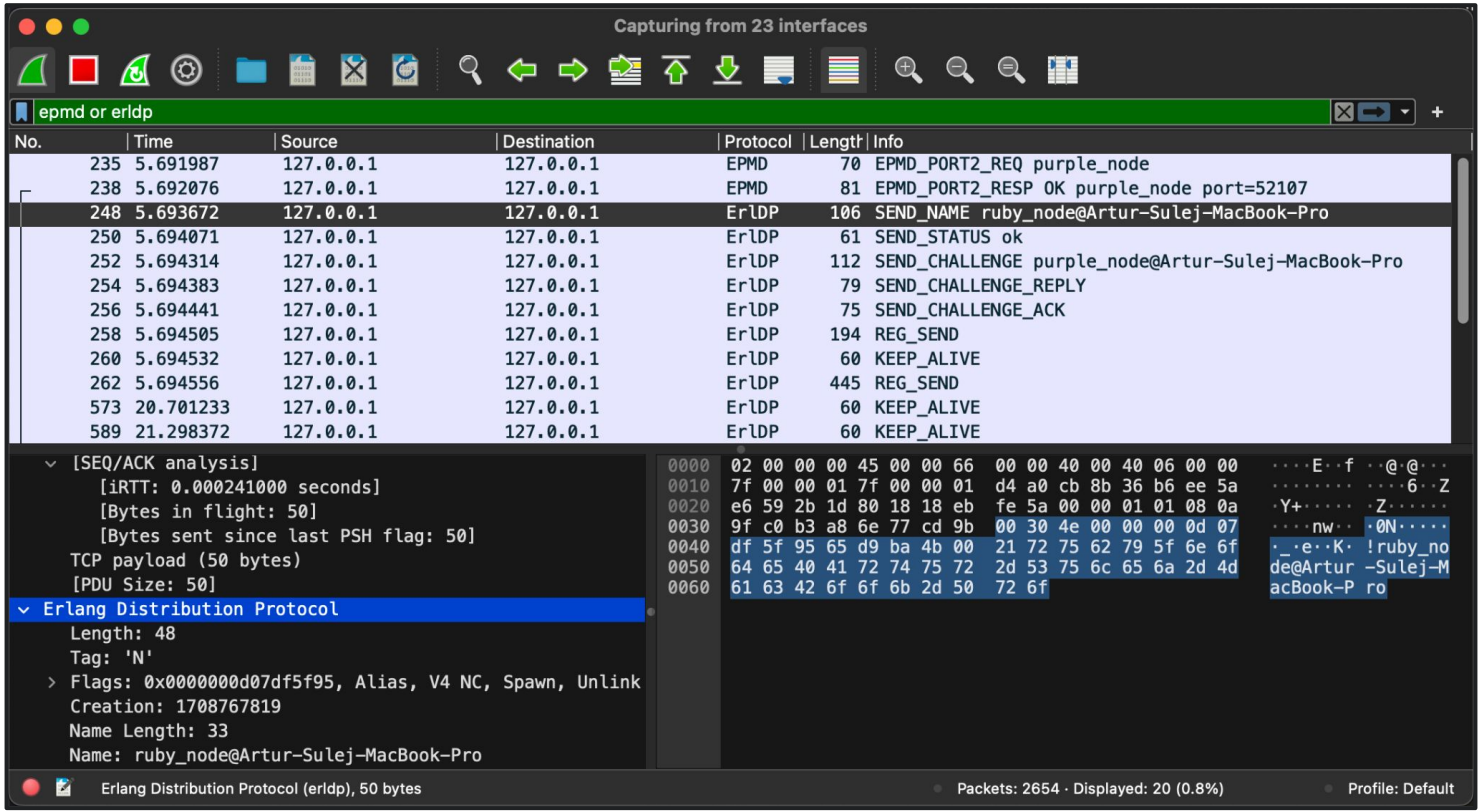


ErlDP

send name

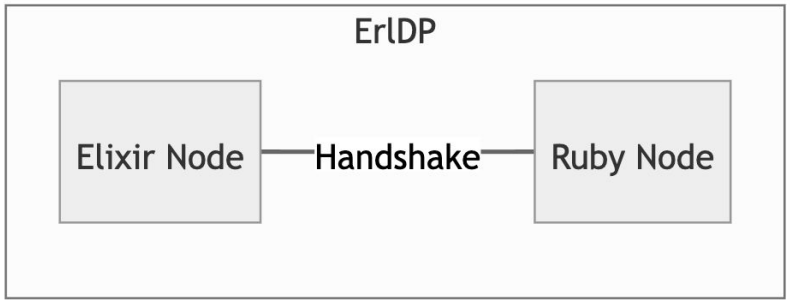
2	1	8	4	2	Nlen
Size	Tag ('N')	Flags	Creation	Nlen	Name

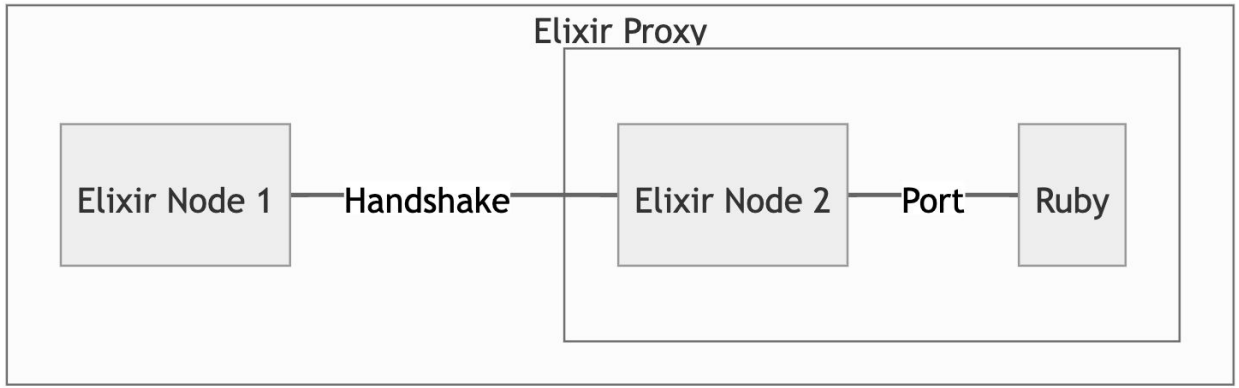
Wireshark


Capturing from 23 interfaces
epmd or erldp
No. Time Source Destination Protocol Length Info
235 5.691987 127.0.0.1 127.0.0.1 EPMD 70 EPMD_PORT2_REQ purple_node
238 5.692076 127.0.0.1 127.0.0.1 EPMD 81 EPMD_PORT2_RESP OK purple_node port=52107
248 5.693672 127.0.0.1 127.0.0.1 ErlDP 106 SEND_NAME ruby_node@Artur-Sulej-MacBook-Pro
250 5.694071 127.0.0.1 127.0.0.1 ErlDP 61 SEND_STATUS ok
252 5.694314 127.0.0.1 127.0.0.1 ErlDP 112 SEND_CHALLENGE purple_node@Artur-Sulej-MacBook-Pro
254 5.694383 127.0.0.1 127.0.0.1 ErlDP 79 SEND_CHALLENGE_REPLY
256 5.694441 127.0.0.1 127.0.0.1 ErlDP 75 SEND_CHALLENGE_ACK
258 5.694505 127.0.0.1 127.0.0.1 ErlDP 194 REG_SEND
260 5.694532 127.0.0.1 127.0.0.1 ErlDP 60 KEEP_ALIVE
262 5.694556 127.0.0.1 127.0.0.1 ErlDP 445 REG_SEND
573 20.701233 127.0.0.1 127.0.0.1 ErlDP 60 KEEP_ALIVE
589 21.298372 127.0.0.1 127.0.0.1 ErlDP 60 KEEP_ALIVE
[SEQ/ACK analysis]
[iRTT: 0.000241000 seconds]
[Bytes in flight: 50]
[Bytes sent since last PSH flag: 50]
TCP payload (50 bytes)
[PDU Size: 50]
Erlang Distribution Protocol
Length: 48
Tag: 'N'
Flags: 0x0000000d07df5f95, Alias, V4 NC, Spawn, Unlink
Creation: 1708767819
Name Length: 33
Name: ruby_node@Artur-Sulej-MacBook-Pro
0000 02 00 00 00 45 00 00 66 00 00 40 00 40 06 00 00
0010 7f 00 00 01 7f 00 00 01 d4 a0 cb 8b 36 b6 ee 5a
0020 e6 59 2b 1d 80 18 18 eb fe 5a 00 00 01 01 08 0a
0030 9f c0 b3 a8 6e 77 cd 9b 00 30 4e 00 00 00 0d 07
0040 df 5f 95 65 d9 ba 4b 00 21 72 75 62 79 5f 6e 6f
0050 64 65 40 41 72 74 75 72 2d 53 75 6c 65 6a 2d 4d
0060 61 63 42 6f 6f 6b 2d 50 72 6f
Erlang Distribution Protocol (erldp), 50 bytes
Packets: 2654 · Displayed: 20 (0.8%)
Profile: Default

Let’s see the code

Ruby ErlDP vs Elixir Proxy





Port

Ports provide a mechanism to start operating system processes (external to the Erlang VM) and communicate with them.

Elixir's `Port` is a convenient, high-level interface to Erlang's `:port` mechanism.

Communication:

- Elixir – sending and receiving messages
- External process – reading `stdin` and writing to `stdout`

Port – simple example

```
port = Port.open({:spawn, "ruby reverse.rb"}, [:binary])
Port.command(port, "No palindromes here!\n")

result =
  receive do
    {^port, {:data, data}} -> data
  after
    2000 -> {:error, :timeout}
  end

IO.puts("Result: #{inspect(result)}")
Port.close(port)
```

Elixir Proxy & Port

How to draw an owl

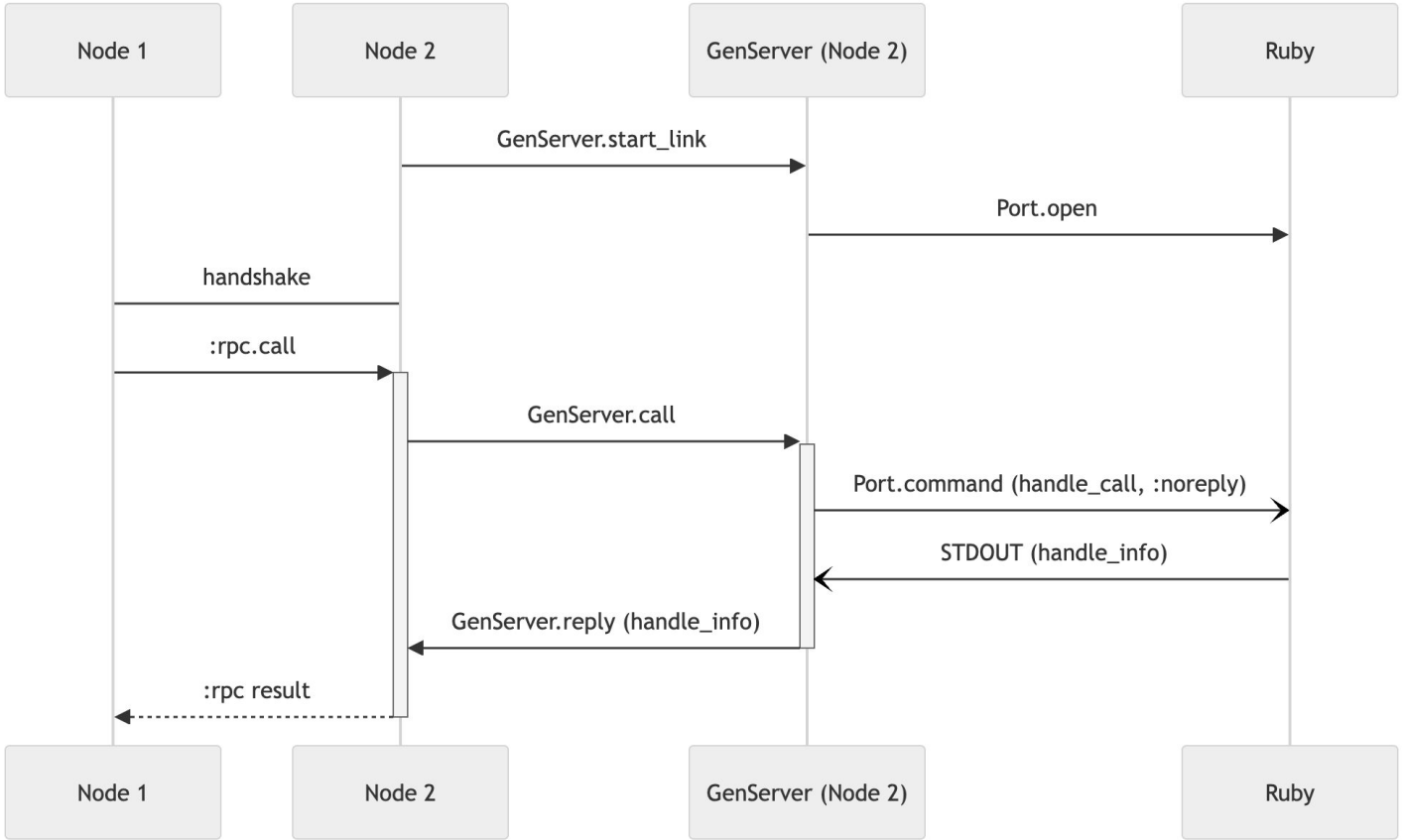
1.

1. Draw some circles

2.

2. Draw the rest of the f***ing owl

Elixir Proxy & Port



Elixir Proxy & Port

- Raw message passing → wrapped in GenServer
- Simple string → JSON
- Matching requests with responses
- Gateway module (called in `:rpc`)
- Mimicking `:rpc` in Ruby
- Single Port → pool (scalability)
- Error handling
- Macro for DSL in Gateway

Let’s see the code

Repo

 https://github.com/Artur-Sulej/ruby_node

Resources

1. https://www.theerlangelist.com/article/outside_elixir
2. https://hexdocs.pm/elixir/1.16.1/Port.html
3. https://www.erlang.org/doc/apps/erts/erl_dist_protocol.html
4. https://learnyousomeerlang.com/distribunomicon
5. https://github.com/CloudI/CloudI/blob/e2ea214941486dfa3fbb437e982c9b95ebb4c72b/src/api/ruby/erlang.rb